BY DONALD W CORSON • EM MICROELECTRONIC

Windowed-watchdog timers enhance system security

A SUPERVISORY FUNCTION ENABLES SYSTEM RECOVERY TO PREVENT EXECUTION ERRORS.

As microprocessor-controlled systems begin to carry out more and more functions involving human safety, the importance of close performance monitoring is increasing. The low cost and range of features for many of today's microprocessor functions allow their use in many applications that were previously the domain of dedicated hardware. Although microprocessors are highly flexible tools, the probability of code errors in their programs lowers their functional reliability. Defensive programming techniques, such as filling unused ROM with halt or illegal instructions to trap illegal jumps in code space, will aid in program debugging. They can also provide a small but useful mechanism for gracious recovery when deployed. But even with the most careful and complete testing, you won't find every error; no method can ensure 100% coverage.

Systems that could cause bodily injury if they malfunction require high reliability. Examples of such systems include automotive antilock-braking or steering systems; medical instruments, such as insulin pumps; robots; industrial-control systems; automatic doors; nuclear-power-plant controls; and avionics. These systems must be able to recover from a crash without human assistance, such as someone pressing a reset button, because such intervention would probably occur too late to prevent injury.

A watchdog timer is a subsystem that can cause a program reset or NMI (nonmaskable interrupt) if a microprocessor does not react within a certain amount of time. In many cases, the timer can catch a misbehaving microprocessor system. For highly sensitive applications, designers should use windowed-watchdog timers, which activate when system code clears them either too slowly or too quickly. Their use adds another class of recognizable program errors or faulty hardware behavior. Ideally, a watchdog-monitored system can restart itself back into a working state without the user even knowing that an error occurred. To achieve this level of comfort, the system and software design must be able to accept a reset at any time and resume normal operation without operator intervention.

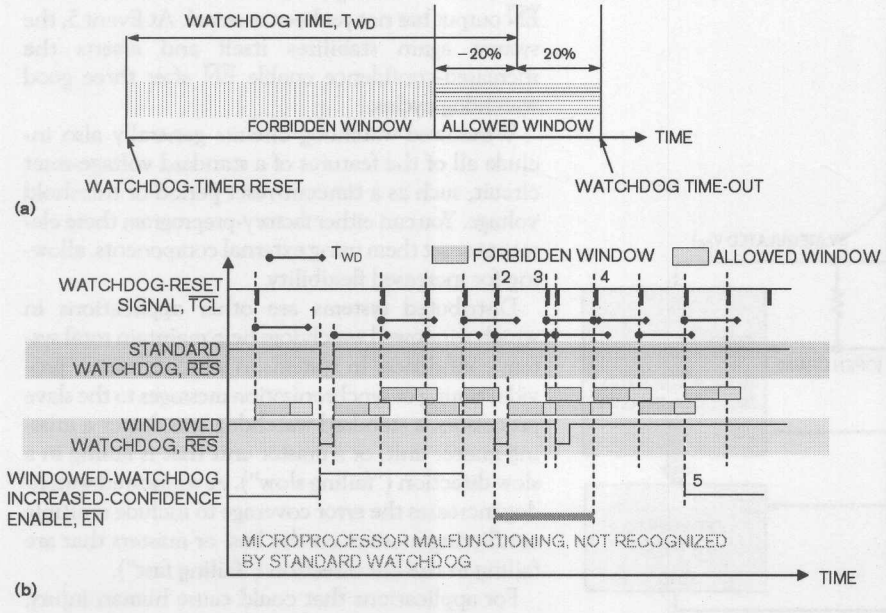

Figure 1 Two distinct periods exist in the timing of a windowed-watchdog timer (a). Comparing windowed-watchdog timers and standard-watchdog timers reveals the effect of the windowing period (b).

Many microcontrollers offer an internal programmable watchdog with similar functions. Software can disable these internal watchdog timers, so they do not provide the same protection for safety-critical applications as do independent external watchdog circuits. Critical applications should employ an external watchdog-reset circuit.

BASIC OPERATION

Standard watchdogs are incrementing counters that set their output when the counters reach their maximum value. The microcontroller must reset the counter by creating a falling edge on the timer's clear input. If the program execution is faulty because of a program error, or if an external disturbance slows the program execution, the counter will reach its maximum value, and the

tion for gating motor signals. For instance, this function can immediately stop the motor movement when a system cannot trust the processor behavior and allow it again only when it is confident that the processor is running properly. The watchdog timer reasserts this signal only after it sees three good $\overline{TCL}$ edges and removes the signal simultaneously with the $\overline{RES}$ output assertion when it detects a processor malfunction.

MORE AT EDN.COM

Go to www.edn.com/ms4131 and click on Feedback Loop to post a comment on this article.

SOFTWARE CONSIDERATIONS HAVE A ROLE

Adding a windowed watchdog to a system is an important step in increasing system confidence, but if the watchdog timer's service routine is a timer-triggered interrupt routine just for this watchdog, it is useless. It is very possible that the entire system could crash, yet the timer-triggered interrupt continues to service the watchdog at the appropriate intervals, indicating that all is well.

Always keep in mind the basic rules of embedded programming. Always fill unused program memory with defined patterns and be sure that this pattern is defined for every possible address in memory where a misguided jump could land. The strategy depends on the processor. You can use multibyte or word instructions where a wayward jump could land in the middle of an address boundary.

In general, use halt instructions or known illegal instructions if the processor core traps them, as either instruction traps illegal jumps regardless of the cause. The halt causes a watchdog to trigger, whereas the trapping and processing action that occurs after an illegal instruction depends on the system architecture. Both techniques are useful in a debugging environment to help trace the cause of the illegal jump. In production units, you can use them to set a reset or to trigger a routine that puts the equipment in a known or safe mode.

NEVER SERVICE A WATCHDOG TIMER USING A ROUTINE SOLELY FOR THAT PURPOSE.

Never service a watchdog timer using a routine solely for that purpose. The only exception to this rule could be in multitasking systems. Because such systems are often nondeterministic, one option for periodically servicing the watchdog timer is to have a monitor task that services the watchdog, depending on clues that other tasks leave. By incrementing counters when they have finished certain processing functions, for example, the system tasks can leave enough information for a monitoring task to decide whether the system is well. Because this approach uses software to take over a hardware-safety function, designers should make sure the system is sufficiently deterministic so that the watchdog-timer service function uses a working routine.

Also include reset-time processor validation in the embedded-system design. Although processor failures are rare and most often catastrophic, partial failures do occur. Processor validation, which you must do in assembly-language code, should begin with a simple unconditional jump command and then continue to all of the commands that the application uses, where the tested commands can find use later in the tests for other commands. Although programmers may not like creating such test code, it can provide considerable system security and even cost savings, because it allows the system to demonstrate that the processor is thoroughly tested, both in production test (possibly eliminating the need for a dedicated testing station) and in application use.

DON'T FORGET THERMAL CONSIDERATIONS

Windowed-watchdog-timer circuits are also available with one built-in LDO (low-dropout regulator) or more on chip. Such circuits are especially useful in decentralized systems, such as automotive and industrial-automation applications, as they can monitor the security and provide the power-supply regulation in one component (**Figure 2**).

As with any voltage regulator, the pc-board layout is important to the success of the design. The routing of the decoupling capacitors to the supply and ground traces or planes must be clean and short. Circuitous paths increase the circuit inductance and possibly the cross-coupling between inputs and outputs. Clean separation between the logic supply and the power portion of the circuitry is especially important in circuits controlling electrical motors, due to the large spikes that they produce on the power-supply lines.

Also, designers should take into account thermal issues when planning the layout. The housing of many ICs containing LDOs has a heat-sink contact called a "thermal slug" that you must solder to the pc board. The pc board should provide adequate surface area so it can function as a radiator around the chip. It is best on both board sides to have circuit planes that connect to the slug using thermal vias, to transfer the heat from the chip as efficiently as possible. The actual thermal resistance in any application depends greatly on the physical configuration of the complete module, pc-board cooling surfaces, thickness, airflow, convection, horizontal or vertical orientation, and other factors.

Watchdog components that can recognize that they are being placed in sleep mode, and so adapt their behavior to reduce system power consumption without decreasing security, are also available. These components suit ultralow-power applications using sleep mode, such as those for CAN-bus communication, in which you can disable functional units under software control. EDN

AUTHOR'S BIOGRAPHY

Don Corson began his career in computer-peripheral development at Philips in Germany. For the last five years, he has been with Swatch Group and EM Microelectronic. Corson was responsible for the battery system for Swatch's hybrid-car project and works at Swatch Group's semiconductor fab and design company, EM Microelectronic, on battery-related low-power, low-voltage projects. You can reach him at dcorson@emmicroelectronic.com.